Un petit convertisseur word vers dokuwiki

Transformation word -> txt

```
antiword -f -w 0 nom_fichier.doc > nom_fichier.txt
```

Transformation txt -> dokuwiki

```
./antiword2dokuwiki.pl nom_fichier.txt > nom_fichier.dokuwiki.txt
```

Code

Petits trucs appris au cours de l'écriture du code ci-dessous

- Échapper les caractères spéciaux d'une variable avant d'utiliser le résultat dans le modèle d'une expression régulière.

usage d'une variable dans une regex perl

```
# la partie importante :
$va_chercher_quote_meta = quotemeta($va_chercher);
if ( $ligne =~ /$va_chercher_quot_meta\s*$/ ) {
    print "C'est bien ! Bon chien-chien !\n";
} else {
    print "Maintenant couché, Maisdors !\n"
}
```

Le script

antiword2wiki.pl

```
#!/usr/bin/perl -w

use warnings;
use strict;

# ce qui suit est *essentiel* au bon fonctionnement
# des expressions régulières utilisées ici.
# Tout doit etre en utf8.
use locale;
use utf8;
use encoding "utf8";

#binmode( STDIN,  ':utf8' );
#binmode( STDOUT, ':utf8' );

#use feature 'unicode_strings';

# variable pour savoir si on génère un sommaire.
my $sommaire = 0;

# variable pour savoir si on génère aussi les numéros de section
my $sections = 0;

# variable pour savoir si on fait du ménage (clean) sur les
# titres de section (pas de formatage : gras, italique) ni de
# ":" en fin de titre.
my $clean_sections = 0;
```

```
my $usage = <<"EOF";
usage: $0 [-h|-t|-c|-n] [<fichier généré par antiword>]*

Résumé:
  Ce script génère du texte conforme aux usages de formatage de dokuwiki à partir d'un
  fichier texte résultat d'une transformation, par antiword, d'un fichier word bien
  formaté.
  Par défaut, le texte à transformer se tappe à l'entrée standard.

Options:

 -h                 Cette aide
 -t                 Génère la table des matières extraite du fichier donné en attribut
 -n                 Génère aussi les numéros de sections dans les sections
 -c                 Génère des noms de section cleans (pas de formatage du texte des sections)
 fichier_antiword   Le fichier à utiliser pour produire un texte conforme à dokuwiki

Présentation:
  script de génération de code dokuwiki à partir d'un
  document texte généré par "antiword -f -m 8859-1 -w 0 nom_fichier_word.doc | iconv -f iso-8859-1 -t utf-8".

  Les options d'antiword sont importantes : -w 0 signifie
  que les paragraphes ne sont pas justifiés mais tiennent sur une
  seule ligne. C'est important pour ce qui concerne les listes
  à puces. Si tel n'était pas le cas, on trouverait des passages
  à la ligne dans les listes entrainant un formatage incorrect
  des sous-listes par exemple.

Format accepté:

== Règles de transformations et présuppositions
  * il y a un sommaire reprenant chaque ligne. Ce sommaire commence
  par la chaine "Contenu du document :" avec un espace final qui
  peut etre insécable ou non et d'éventuels espaces après les ":".

  * Ce sommaire servira à construire la table des matières soit un
  tableau des différents titres de section et sous-section. Une fois ce
  tableau établi, les titres sont recherchés dans la page les uns après
  les autres. Ils sont remplacés par un formatage wiki correct en
  fonction de la profondeur de section (on se base sur le format de
  numérotation à savoir : <chiffres>[.<chiffres>]*. Le nombre de "."
  indique donc la profondeur de section.
  * Enfin, comme dans tout sommaire, en correspondance du titre et de
  la numérotation de la section, il doit y avoir un numéro de page.
  Ce numéro est obligatoire pour permettre à l'algorithme utilisé
  de fonctionner.

  En dokuwiki, nous avons cela :
  * h1 ====== ====== Correspond au titre principal (non utilisé sauf
  pour le titre mais celui-ci n'est pas reconnu par la procédure).
  * h2 ===== ===== Les titres de sections sont ici
  * h3 ==== ==== Titres des sous-sections (celles qui ont un "." dans
  leur numérotation).
  * etc.

  Les titres de section doivent etre encadrés par une ligne vide (ne
  contenant éventuellement que des espaces) de part et d'autre du
  titre lui-meme.

=== Exemple de format d'entrée accepté

>>> Début document exemple <<<

  Plein de choses avant. Traité seulement pour les apparitions de
  - début liste à puce
  /toto/ en italique
  *gras* en gras
  _souligné_
```

```
  Contenu du document :



  1 Introduction        1
  2 Présentation        2
  2.1 Généralités       4
  2.2 Détails           8
  2.2.1 Aspects particuliers 15
  3 Conclusion         20


  1 Introduction

  ipsum temporis megna et terra optemporum _isque_

  2 ...
  etc

>>> Fin document exemple <<<

=== Sortie de l'exemple court précédent

>>> Début document généré <<<

Plein de choses avant. Traité seulement pour les apparitions de
  * début liste à puce
//toto// en italique
**gras** en gras
_souligné_

===== Introduction =====

ipsum temporis megna et terra optemporum __isque__

===== ... =====
etc

>>> Fin document généré <<<

=== Limitations :

  Le titre principal (h1 correspondant à ====== ======) n'est pas
  modifié pour etre formaté correctement au format wiki.

EOF

# traitement de la ligne de commande.
if (scalar(@ARGV) >= 1) {
    if ($ARGV[0] eq "-t") {
        $sommaire = 1;
    } elsif ($ARGV[0] eq "-h") {
        print $usage;
        exit 0;
    } elsif ($ARGV[0] eq "-n") {
        $sections = 1;
    } elsif ($ARGV[0] eq "-c") {
        $clean_sections = 1;
    }
    shift(@ARGV);
}

# fonction syntaxe_dokuwiki:
#
# arguments attendus :
#   - un seul : une chaine de caractères.
#
# fonction :
#   - rend une chaine de caractère qui est conforme au
#   formatage de dokuwiki
#
```

```
sub syntaxe_dokuwiki {

    my $ligne = shift;

    # on trim à gauche seulement - avant le reste
    $ligne =~ s/^\s+//g;
    # remplacement des /italiques/ par //italiques//
    $ligne =~ s|/([^/]+)/|//$1//|g if m|/[^/]+/|;
    # remplacement des /en gras/ par //en gras//
    $ligne =~ s|\*([^*]+)\*|**$1**|g if m|\*[^\*]+\*|;
    # remplacement des _soulignés_ par __soulignés__
    $ligne =~ s|_([^_]+)_|__$1__|g;
    # remplacement des "-" en début de ligne par "  *"
    $ligne =~ s|^\s*- |  * |g;
    # remplacement des "o" en début de ligne par "    *"
    $ligne =~ s|^\s*o |    * |g;
    # remplacement des "" (utf8 : \xe2\x96\xaa) en début de ligne par "      *"
    # my $carre = "\xe2\x96\xaa";
    # avec la nouvelle méthode utilisant antiword latin1 + iconv , ce ne sont plus
    # des carrés mais des "." simples.
    my $carre = quotemeta(".");
    $ligne =~ s|^\s*$carre |      * |g;

    return $ligne;
}

# fonction creer_section_wiki:
#
# variables:
# - titre: le titre qu'il faut mettre avec le bon formatage
# - niveau : le niveau de profondeur du titre
#
sub creer_section_wiki {
    my $titre = shift;
    my $niveau = shift;
    my $nsection = shift;

    my $section = "";
    for (my $k = 1 ; $k <= 6 - $niveau; $k++) {
        $section .= "=";
    }
    return "$section " . ($nsection ? "$nsection - " : "") . $titre . " $section\n";
}

# == Variables

# le tableau des indexes du sommaire
my @index;

# $encadre_titre : variable permettant de savoir si un titre est entouré,
#   de part et d'autre, par deux lignes ne comportant éventuellement que des
#   espaces ou simplement vide.
#   valeurs :
#     * 0 : aucune ligne vide détectée
#     * 1 : une ligne vide détectée
#     * 2 : ligne vide + titre
#     * 3 : ligne vide + titre + ligne vide
my $encadre_titre = 0;

# $titre_trouve : valeur booléenne qui indique si un titre du sommaire
#   est trouvé sur la ligne courante traitée.
my $titre_trouve = 0;

# $sommaire_traite : valeur booléenne qui indique si le sommaire a été
#   traité et si le tableau des entrées de ce sommaire est constitué.
my $sommaire_traite = 0;

# $debut_sommaire : valeur booléenne qui indique si on entre dans le
#   début du sommaire - à savoir : si "Contenu du document :" est passé
#   *et* si on est dans les lignes vides séparant ce texte du contenu
#   véritable du sommaire.
```

```
my $debut_sommaire = 0;

# $i : index du tableau des titres des sections et sous-sections.
my $i = 0;

# $j : index courant du titre à chercher dans le tableau @index.
my $j = 0;
while (<>) {
    chomp;

    # print "[debug] - ligne = '$_'\n";
    # == génération du sommaire.
    #
    # attention, il y a un espace insécable avant les ":" - couvert par \h et pas \s.
    if (m!^(?:\s| |[*/_])*Contenu du document(?:\s| ):(?:\s| |[*/_])*$!) {
        # print "[debug] - entrée dans contenu du document\n";
        # on entre dans le début du sommaire
        $debut_sommaire = 1;
        # On avance dans la lecture du fichier
        while (<>) {
            # print "[debug] - ligne = '$_'\n";
            # si on est à peine rentré dans le sommaire,
            # - on n'en sort pas avec une succession de lignes vides
            # - on peut en sortir après avoir eu au moins une ligne
            #   contenant quelque chose.
            if (/^\s*$/ && $debut_sommaire == 1) {
                next;
            } elsif (! /^\s*$/ && $debut_sommaire == 1) {
                $debut_sommaire = 0;
            } else {
                $debut_sommaire = 0;
            }

            # si ligne vide, on quitte car toutes les sections du sommaire ont été traitées
            last if (/^\s*$/);

            # Cas général :
            # - dans ce cas, toutes les lignes suivantes qui ne sont pas
            #   des lignes vides sont des éléments de titre
            # Le format de ces lignes est du type :
            #   - espaces suivi de numéros de sections et sous-sections séparées par des "."
            #   et finissant par des espaces avec un nombre terminant la ligne.
            if (/^\s*(\d+(\.\d+){0,})\s+(.*)\d+$/) {
                my $num_section = $1;
                # traitement de la section (combien de sous sections)
                # le niveau de sous-section est le nombre de "." dans le numéro de section.
                my $niveau = scalar(split(/\./, "$num_section"));

                # on trim le titre (suppression des espaces avant et après)
                my $titre = $3;
                $titre =~ s/^\s+|\s+$//g;
                # impression du tableau dans le cas où le sommaire est demandé (-t)
                print "$niveau|$titre|$num_section\n" if $sommaire;

                # insertion du titre et du niveau dans un tableau
                # contenant tout le sommaire.
                $index[$i]{"niveau"} = $niveau;
                $index[$i]{"titre"}  = $titre;
                $index[$i]{"numero"} = $num_section;
                $i++;
            }
            # fin du traitement du sommaire
            $sommaire_traite = 1;
        }
    } else {
        # print "[debug] - encadre_titre courant = $encadre_titre\n";

        # == traitement de la génération du document

        # traitement des sections une par une (index de sous-section courante = $j)
        if ($sommaire_traite) {
```

```
            my $titre_a_chercher = $index[$j]{"titre"};

            #print "[debug] - nouveau titre à chercher = '$titre_a_chercher'\n";

            # ligne vide et précédée par une ligne vide ou une pleine
            if (/^\s*$/ && ($encadre_titre == 0 || $encadre_titre == 1)) {
                $encadre_titre = 1;
                print "$_\n" if !$sommaire;
                next;
            # ligne non vide et précédée par une ligne vide
            } elsif (!/^\s*$/ && $encadre_titre == 1) {
                $encadre_titre = 2;

                # print "[debug] - titre à chercher : '$titre_a_chercher'\n";
                # print "[debug] - chaine = '$_'\n";

                # Le titre à chercher ne doit pas etre vide et il doit voir les
                # caractères spéciaux qu'il contient échappés pour ne pas avoir
                # de problèmes dans la recherche du modèle.
                # Remarque : utiliser une recherche de chaîne de caractères précise
                # (substring) plutôt que d'utiliser les expressions régulières
                # serait plus sûr et efficace mais peut-être aussi plus juste
                # sémantiquement.
                my $titre_quote_meta = quotemeta($titre_a_chercher) if $titre_a_chercher;


                # remarque importante sur le titre :
                # dans le sommaire, il n'y a pas de formatage (gras/italique/souligné)
                # alors qu'il peut en exister dans le titre réel, dans le corps
                # du document. Pour chaque ligne potentiellement «titre», il faut
                # commencer par supprimer le formatage.
                # $ligne_traitee : la ligne moins les caractères */_
                my $ligne_traitee = $_;
                $ligne_traitee =~ s|[_/*]||g;
                # expression régulière compliquée pour les titres
                # - espaces possibles suivis de chiffres (possible) suivis d'espaces
                #   possibles ou des caractères de mise en forme /, * ou _ (possible)
                #   suivi par le titre suivi par des espaces ou /, *, _ (possible).
                if ($titre_a_chercher && ($ligne_traitee =~ m!^(?:\s| )*([/*_]*)\d*(?:\s| )*([*/_]*)
($titre_quote_meta)((?:\s| )*)([*/_]*)(?:\s| )*$!)) {

                    # print "[debug] - titre_quote_meta = '$titre_quote_meta'\n";
                    # print "[debug] - ligne brute = '$_'\n";

                    my $ligne = "$1$2$3$4$5";
                    # print "[debug] - ligne = '$ligne'\n";
                    $titre_trouve = 1;
                    # print "[debug] - titre trouvé\n";

                    my $ligne_suivante = <>;
                    chomp($ligne_suivante);
                    if (!$ligne_suivante =~ /^\s*$/) {
                        print "[debug] - encadre_titre = $encadre_titre ; titre_a_chercher =
$titre_a_chercher\n";
                        print "[debug] - ERROR - indication erreur : la ligne suivant le titre n'est pas une
ligne vide ! Bizare !\nligne ='$_'\n";
                        $encadre_titre = 0;
                    } else {
                        # print "[debug] - La ligne suivante est bien une ligne vide.\n";
                        # print "[debug] - La ligne qui fait office de titre est : '$ligne'.\n";

                        # computation du titre au format wiki
                        # On passe plutôt la ligne que le titre car les titres
                        # peuvent avoir du formatage particulier en
                        # plus de celui qui apparaît dans le sommaire.
                        if ($clean_sections) {
                            $titre_a_chercher =~ s/^(.*?)(?: |\s)*:(?: |\s)*$/$1/;
                            $_ = &creer_section_wiki($titre_a_chercher, $index[$j]{"niveau"}, $sections ? $index
[$j]{"numero"} : "");
                        } else {
                            $_ = &creer_section_wiki($ligne, $index[$j]{"niveau"}, $sections ? $index[$j]
```

```
{"numero"} : "");
                        }
                        # on a eu une ligne suivante vide, donc il faut le dire
                        $encadre_titre = 1;
                    }
                    $j++; # pour passer au prochain titre de section à chercher.
                } else {
                    $titre_trouve = 0;
                    $encadre_titre = 0;
                }
            } elsif (/^\s*$/ && $encadre_titre == 2) {
                $encadre_titre = 3;

                # print "[debug] - bon modèle de titre : ligne vide + ligne pleine + ligne vide = $_\n" if
!$sommaire;

                print "$_\n" if !$sommaire;
                next;
            } else {
                $encadre_titre = 0;
            }

        }
        my $ligne = &syntaxe_dokuwiki($_);

        # si on ne veut pas afficher que le sommaire :
        print "$ligne\n" if !$sommaire;
    }
}
```

Code de l'interface web

convertisseur_word2wiki.php

```
<?php
// entetes et locales.
// setlocale(LC_CTYPE, "en_US.UTF-8");
header('Content-Type: text/html; charset=UTF-8');

?>
<html>
<head>
    <title>Convertisseur limité de document word en syntaxe dokuwiki</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

<div>
<h1 align="center">Convertisseur simple d'un fichier word en un texte de syntaxe wiki</h1>
</div>
<?php

if (isset($_POST['MAX_FILE_SIZE']) && isset($_POST['deposer']) && isset($_FILES['fichier_word'])) {
    if ($_POST['MAX_FILE_SIZE'] > 1000000) {
        print "[Erreur] - taille du fichier supérieure à 1 000 000 d'octets.<br />\n";
    } else {
        // traitement
        $liste_fichiers_utilisables = array(
                                            '41326ef3286265fd2be4d2958b1f8820',
                                            'd724f5a114680faa8abe9975a5295670',
                                            '870d5770f8b4828c453290f2d8ee4927',
                                            'ef7d4f8707e273c63c0c695fea9e2c93',
                                            '999d84e7eb43c1c987f6b25faf669d49',
                                            '12fe72d929d401d9826d249b8322a7d9',
                                            'd9ff4859054165adeae3f4be71097921',
                                            '431148a540c5d7880d456241d8c914ad',
                                            '6fb7493bef4ca2534a595063ce54e912',
```

```
                                        'fb8d6c7c94c64457b98f6470f517c888',
                                       );
        $filename = '';
        foreach ($liste_fichiers_utilisables as $f) {
           if (file_exists("/tmp/$f.doc")) {

continue;
            } else
{
                $filename = "/tmp/$f.doc";
                break;
            }
        }

        if ($filename == '') {
            print "[Erreur] - il n'y a plus de slot disponible pour faire le traitement de votre de fichier.<br
/>\n";
            print "[Info] - Seuls " . sizeof($liste_fichiers_utilisables) . " fichiers peuvent etre traités
concurrentiellement par le serveur.<br />\n";
        } else {
            // déplacement du fichier vers sa destination
            if(move_uploaded_file($_FILES['fichier_word']['tmp_name'], $filename)) {
                print "[Info] - SUCCÈS - Le fichier a été déplacé avec succès de " . $_FILES['fichier_word']
['tmp_name'] . " vers $filename<br />\n";

                // traitement des options cochées (checkboxes) :
                $antiword = 0; // génération uniquement par antiword ? 0 = non ; 1 = oui
                $options = "";
                if ( $_POST["sommaire"] == "yes" ) {
                    $options = " -t ";
                } elseif ( $_POST["numeros_sections"] == "yes" ) {
                    $options = " -n ";
                } elseif ( $_POST["clean_sections"] == "yes" ) {
                    $options = " -c ";
                } elseif ( $_POST["antiword_brut"] == "yes" ) {
                    $antiword = 1;
                }

                // traitement véritable
                // $arguments = escapeshellargs();
                // Le script perl traitant le fichier antiword !
                $antiword2wiki = '/var/www/sites/www.perso.uhp-nancy.fr/zimmermann/antiword2wiki.pl';
                if ($antiword) {
                    $a_executer = escapeshellcmd("/usr/bin/antiword -m 8859-1 -f -w 0 $filename") . "| iconv -f
iso-8859-1 -t utf-8";
                } else {
                    $a_executer = escapeshellcmd("/usr/bin/antiword -m 8859-1 -f -w 0 $filename");
                    $a_executer .= "| iconv -f iso-8859-1 -t utf-8 | $antiword2wiki $options -";
                }
                print "[Info] - Commande à exécuter : '$a_executer'<br />\n";
                // $derniere_ligne = system($a_executer, $retour_execution);
                $derniere_ligne = exec($a_executer, $sortie_standard, $retour_execution);
                // resultat à afficher dans un textarea non modifiable
                $resultat = "";
                foreach ($sortie_standard as $l) {
                    $resultat .= $l . "\n";
                }
                // print "resultat avant : '$resultat'";
                // $resultat = htmlentities($resultat, ENT_NOQUOTES, 'UTF-8');
                // print "resultat après : '$resultat'";
                // $resultat = utf8_decode($resultat);
                print <<<EOF
<div align="center">
<textarea name="resultat" rows="50" cols="100">$resultat</textarea>
</div>
EOF;
                // un retour correct d'une commande, c'est 0 (d'où le !)
                if (! $retour_execution) {
                    print "[Info] - SUCCÈS à l'exécution ! - retour = $retour_execution<br />\n";
                } else {
                    print "[Erreur] - ÉCHEC à l'exécution ! - retour = $retour_execution<br />\n";
```

```
                }
                // maintenant que le traitement est terminé, il faut supprimer le fichier déplacé pour libérer
                // un slot
                if (unlink($filename)) {
                    print "[Info] - SUCCÈS - suppression du fichier $filename effective !<br />\n";
                } else {
                    print "[Erreur] - ÉCHEC - suppression du fichier $filename impossible !<br />\n";
                }
            } else {
                print "[Erreur] - ÉCHEC - Le fichier n'a pu etre déplacé de " . $_FILES['fichier_word']
['tmp_name'] . " vers $filename<br />\n";
            }
        }
    }

} else {
    print "[Debug] - ";
    print_r($_POST);
    print "name = " . $_FILES['fichier_word']['tmp_name'] . "<br />\n";
    print "name = " . $_FILES['fichier_word']['name'] . "<br />\n";
}

// dans tous les cas : affichage de la form
?>

<form action="<? echo $PHP_SELF; ?>" enctype="multipart/form-data" method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
<fieldset>
<table><tr>
<td><label for="fichier_word">Fichier Word au bon format (Cf. documentation ci-dessus)</label></td>
<td><input type="file" name="fichier_word" /></td>
</tr>
<tr>
<td>Taille max : <b>1 million d'octets</b></td>
<td><input type="submit" name="deposer" value="Convertir le fichier !" /></td>
</tr>
<tr>
<td><label for="sommaire">Générer le sommaire uniquement</label></td>
<td><input type="checkbox" name="sommaire" value="yes" /></td>
</tr>
<tr>
<td><label for="numeros_sections">Générer les titres avec les numéros de sections</label></td>
<td><input type="checkbox" name="numeros_sections" value="yes" /></td>
</tr>
<tr>
<td><label for="clean_sections">Générer des sections épurées du formatage et des ":" finaux</label></td>
<td><input type="checkbox" name="clean_sections" value="yes" /></td>
</tr>
<tr>
<td><label for="antiword_brut">Générer seulement la conversion word vers texte faite par antiword</label></td>
<td><input type="checkbox" name="antiword_brut" value="yes" /></td>
</tr>
</table>
</fieldset>

</form>
<!-- nb de slots disponibles : 10

41326ef3286265fd2be4d2958b1f8820  -
d724f5a114680faa8abe9975a5295670  -
870d5770f8b4828c453290f2d8ee4927  -
ef7d4f8707e273c63c0c695fea9e2c93  -
999d84e7eb43c1c987f6b25faf669d49  -
12fe72d929d401d9826d249b8322a7d9  -
d9ff4859054165adeae3f4be71097921  -
431148a540c5d7880d456241d8c914ad  -
6fb7493bef4ca2534a595063ce54e912  -
fb8d6c7c94c64457b98f6470f517c888

Générés avec :
for i in $(seq 1 10) ; do echo $(for i in $(seq 1 500) ; do perl -e 'srand; rand($.) < 1 && ($line = $_) while
```

```
<>; print $line' /usr/share/dict/cracklib-small; done)|md5sum ; done


-->
<hr />
<h2>Documentation</h2>
<pre>
<?php
    $sortie_standard = array();
    #if (is_resource(STDOUT)) fclose(STDOUT);
    #if (is_resource(STDERR)) fclose(STDERR);

    $derniere_ligne = exec("/var/www/sites/www.perso.uhp-nancy.fr/zimmermann/antiword2wiki.pl -h",
$sortie_standard, $retour_execution);

    $resultat = "";
    foreach ($sortie_standard as $l) {
        // il ne faut pas faire (car on est déjà en utf8 et ça ferait merder les choses.
        // $resultat .= utf8_encode($l) . "\n";
        $resultat .= $l . "\n";
    }
    print htmlentities($resultat, ENT_NOQUOTES, 'UTF-8');
    if (! $retour_execution) {
        print "[Info] - SUCCÈS à l'exécution ! - retour = $retour_execution<br />\n";
    } else {
        print "[Erreur] - ÉCHEC à l'exécution ! - retour = $retour_execution<br />\n";
    }
?>
</pre>

</body>
</html>
```